A Simple Post-Quantum Non-Interactive Zero-Knowledge Proof from Garbled Circuits

Hongrui Cui[1] and Kaiyi Zhang[1]

Department of Computer Science, Shanghai Jiao Tong University, 200240 Shanghai
{rickfreeman,kzoacn}@sjtu.edu.cn

Abstract. We construct a simple public-coin zero-knowledge proof system solely based on symmetric primitives, from which we can apply the Fiat-Shamir heuristic to make it non-interactive. Our construction can be regarded as a simplified cut-and-choose-based malicious secure two-party computation for the zero-knowledge functionality. Our protocol is suitable for pedagogical purpose for its simplicity (code is only 728 lines).

Keywords: Zero-knowledge · Garbled circuit · Post-Quantum

1 Introduction

Zero-knowledge proof (ZK) is a fundamental cryptographic primitive which allows a prover to convince a verifier of the membership of an instance x in any $\mathcal{NP}$ language without revealing any information about its witness w [14]. Due to its theoretical significance and practical applications (e.g., the use of ZK in nuclear disarmament [13]), ZK is also a central topic in cryptographic research. In particular, a rich body of research [21,25,16,11,15,7,27,30] (and many others) has successfully constructed *succinct* argument systems with proof size and verification complexity sub-linear in the size of the statement.

Despite the achievements, many current efficient ZK protocols have high prover complexity in order to facilitate the succinct property. While this captures the performance requirement in most cases, we argue that a lightweight prover is crucial for some applications as well. Imagine the following scenario: An computationally weak IoT device captures possibly sensitive data (e.g., biometric) from its sensors and need to prove some properties on the captured data to a powerful server. Due to the privacy requirement, it cannot simply send the data to the server. In this case, a zero-knowledge protocol that has low prover's complexity would be ideal.

1.1 Our Construction

Here we briefly introduce the intuitions of our construction in Figure 1. We consider ZK as a special case of malicious secure two-party computation. Therefore we try to optimize cut-and-choose-based 2PC by utilizing features specific to the zero-knowledge setting.

```
**Prove**$(1^\kappa, C_x, w)$

for $i := 1$ to $2\kappa$ do
    $(GC_i, e_i, d_i) \leftarrow \mathsf{Gb}(1^\kappa, C_x)$
endfor
$b := H(\{GC_i, d_i\})$
$z := \{\}$
for $i := 1$ to $2\kappa$ do
    if $b_i = 0$ then
        $z \leftarrow z \cup e_i$
    elseif $b_i = 1$ then
        $W_i := \mathsf{En}(e_i, w)$
        $z \leftarrow z \cup W_i$
    fi
endfor
return $\pi = (\{GC_i, d_i\}, z)$
```

```
**Verify**$(1^\kappa, C_x, \pi)$

$b := H(\{GC_i, d_i\})$
for $i := 1$ to $2\kappa$ do
    if $b_i = 0$ then
        if $\mathsf{Ve}(GC_i, e_i, d_i, C_x) = 0$ then
            return false
        fi
    fi
    if $b_i = 1$ then
        if $\mathsf{De}(d_i, \mathsf{Ev}(GC_i, W_i)) = 0$ then
            return false
        fi
    fi
endfor
return true
```

Fig. 1. Our post quantum NIZK in a nutshell, where H is a random oracle, $\mathcal{G} = (\mathsf{Gb}, \mathsf{En}, \mathsf{De}, \mathsf{Ev}, \mathsf{ev}, \mathsf{Ve})$ is a garbling scheme. κ is the security parameter.

Informally, a cut-and-choose-based 2PC consists of those steps: Firstly, the garbler generates garbled circuits and sends them to the evaluator. The evaluator randomly chooses a subset of garbled circuits and asks the garbler to open them. On receiving the seeds, the evaluator checks those garbled circuits are generated correctly. Then the two parties execute some oblivious transfers (OT) in order to let the evaluator obtain the garbled label corresponding to his private input. The evaluator also needs additional operations to make sure the garbler's inputs are consistent. Finally, the evaluator accepts the majority outcome of garbled circuits.

Let us try to apply this protocol to zero-knowledge proof directly. Unlike previous constructions [20], we let the prover and verifier be the garbler and evaluator respectively. This change gives rise to several advantages:

1. We no longer need OT because the verifier has no private input. This eliminates public key encryption and leaves only symmetric primitives. We can also get rid of the selective failure attack.
2. We do not have to check the input consistency since the ZK property does not require multiple witnesses to be the same as long as they are valid.
3. We can accept if and only if all outputs are the same instead of accepting the majority. In a 2PC setting, choosing the majority is to avoid one-bit leakage from abortion, however, the verifier is not required to prevent this attack because he has no private inputs.

4. Finally we can see that the protocol is public-coin, i.e., the verifier has no private randomness in this protocol. Therefore we can apply Fiat-Shamir heuristic to this ZK protocol to make it non-interactive.

1.2 Related Works

Here we only briefly review those works related to universal post-quantum non-interactive zero-knowledge proof systems.

zk-STARK This protocol was proposed and first realized by Ben-Sasson et al. in [5]. zk-STARK offers universal, transparent, scalable and post-quantum secure zero-knowledge proof system in the interactive oracle proofs (IOP) model.

MPC-in-the-head Ishai et al. [19] first introduce the "MPC-in-the-head" paradigm from which one can construct a ZK from the black-box use of a secure multiparty computation protocol. Later, this approach was first implemented by ZKBoo [12]. Some subsequent works [1,9] also follow this paradigm.

Garbled Circuit Garbled Circuit (GC) is a cryptographic tool which is widely used in secure multiparty computation. It was invented by Yao [28] in 1986. Recent years, garbled circuits are improved quickly, like point-and-permute [3], row-reduction [24], free-XOR [22], half gate [29] and stacked garbling [18].

ZK from GC Jawurek et al. proposed the first ZK protocol based on garbled circuits [20]. Their construction focuses on the advantage of garbled circuits, namely its efficiency at evaluating non-algebraic functions (e.g., a circuit for block cipher), and thus achieves good performance. Nevertheless, their protocol is not public-coin and cannot be made non-interactive using the standard Fiat-Shamir transformation.

2 Preliminaries

A negligible function, denoted by $\mathsf{negl}(n)$, represents a function $f : \mathbb{N} \to \mathbb{R}$ that for any constant c, there exists an integer N such that for all $n > N$, $f(n) \leq n^{-c}$. We also use $\mathsf{poly}(n)$ to denote *some* polynomial. For integer $n \in \mathbb{N}$ let $[n]$ to denote the set $\{1, 2, ..., n\}$. We use the common definitions of computational and statistical indistinguishable distribution ensembles. Throughout this work, we use κ to denote security parameters. We use PPT to indicate probabilistic polynomial-time and sometimes use the term "efficient" and PPT interchangeably.

An $\mathcal{NP}$ relation $\mathcal{R}$ is defined by a circuit family $\{C_i\}_{i\in\mathbb{N}}$ whose size is bounded by some polynomial. An instance-witness pair (x, w) is included in the relation iff. $C_{|x|}(x, w) = 1$. We use $L(\mathcal{R})$ to denote the language induced by the relation, i.e., $L(\mathcal{R}) = \{x|\exists w, (x, w) \in \mathcal{R}\}$.

2.1 Zero Knowledge Proof

We follow the standard definition of zero knowledge [17], which we recall as follows.

Definition 1. *A protocol π is a sigma protocol for relation R if it's a three-round public-coin protocol satisfying the following three properties:*

Completeness. *If P and V follow the protocol on public input x and private input w where $(x, w) \in \mathcal{R}$ then V always accepts.*

Special soundness. *There exists an efficient PPT algorithm A such that given any x and any pair of accepting transcript (a, e, z) and (a, e', z') for x where $e \neq e'$ extracts w such that $(x, w) \in \mathcal{R}$.*

Honest-verifier zero-knowledge. *There exists a PPT simulator S which on input x, e generates a transcript (a, e, z) such that for any $(x, w) \in \mathcal{R}$ the transcript is identically distributed as in the real execution.*

2.2 Garbled Circuit

We follow the definition of garbled circuit in [20], which is derived from the standard definitions [4]. We first explain the syntax of a garbling scheme and then list the properties of a garbling scheme that is required in this paper.

Definition 2. *A garbling scheme is defined by a tuple $\mathcal{G} = (\mathsf{Gb}, \mathsf{En}, \mathsf{De}, \mathsf{Ev}, \mathsf{ev}, \mathsf{Ve})$:*

- *The garbled circuit generation function Gb is a randomized algorithm that on input a security parameter 1^κ and the description of a Boolean function $C : \{0,1\}^n \to \{0,1\}$, outputs a triple of strings (GC, e, d).*
- *The plaintext evaluation algorithm ev evaluates the function described by C i.e., $\mathsf{ev}(C, w) = C(w)$.*
- *The encoding function En is a deterministic function that uses e to map an input w to a garbled input W.*
- *The garbled evaluation function Ev is a deterministic function that evaluates a garbled circuit GC on an encoded input W to get an encoded output Z.*
- *The decoding function De, using the string d, decodes the encoded output Z into a plaintext output z.*
- *In addition to the standard algorithms, a verifiable garbled scheme has an extra procedure Ve that, on input garbled circuit GC, a description of a Boolean function C, and the encoding information e, outputs 1 (accept) or 0 (reject).*

We require the following standard properties of a garbling scheme.

Definition 3 (Correctness). *Let $\mathcal{G}$ be a garbling scheme described as above. We say that $\mathcal{G}$ enjoys correctness if for all $C : \{0,1\}^n \to \{0,1\}, w \in \{0,1\}^n$ such that $C(w) = 1$, the following probablity is negligible in paramter κ:*

$$\Pr[(GC, e, d) \leftarrow \mathsf{Gb}(1^\kappa, C), W \leftarrow \mathsf{En}(e, w) : \mathsf{De}(d, \mathsf{Ev}(GC, W)) \neq 1]$$

Definition 4 (Privacy). *Let $\mathcal{G}$ be a garbling scheme described as above. We say that $\mathcal{G}$ enjoys privacy if for all $C : \{0,1\}^n \to \{0,1\}$ there exists a PPT algorithm $\mathsf{Gb.Sim}$ such that given plaintext output y, generates the garbled circuit, decoding information, and garbled input that is indistinguishable from a real execution. In particular, the following two distributions are computationally indistinguishable for any input w.*

- $\{(GC, e, d) \leftarrow \mathsf{Gb}(1^\kappa, C), W \leftarrow \mathsf{En}(e, w) : (GC, d, W)\}$
- $\{y \leftarrow \mathsf{ev}(C, w), (GC, d, W) \leftarrow \mathsf{Gb.Sim}(C, y) : (GC, d, W)\}$

Remark 1. In order to facilitate the security proof of our construction, we require the following two additional requirements on the encoding function $\mathsf{En}(e, w)$.

Projective. Fixing s, suppose the function En maps an n-bit string to an $n\ell$-bit one, then the map $f : w \mapsto \mathsf{En}(e, w)$ can be "decomposed" into n functions $f_1, ..., f_n$ such that $f(x) = f_1(w_1), ..., f_n(w_n)$ where w_i is the i^{th} bit of w.

Injective. Let e be generated from $(GC, e, d) \leftarrow \mathsf{Gb}(C)$, the map $f : w \mapsto \mathsf{En}(e, w)$ is injective with high probability over the randomness of Gb.

The first property is standard [4] and is closely related to the application of garbled circuits in secure two-party computation. The second one is naturally satisfied by some natural constructions. For example, let $W = \mathsf{PRF}(s, w)$ where PRF is a pseudorandom function with a large enough range, then the probability of a collision is negligible.

The two properties listed above facilitates an efficient extraction procedure $\mathsf{Gb.Ext}$ that outputs w given e and encoded input W — the projective property allows us to extract bit-by-bit while the injective property guarantees the uniqueness of the extraction. In particular, for every input position i, the extractor test whether the output block corresponds to 0 or 1 and sets the results accordingly, as shown in Figure 2.

Finally, we require the following verifiability property of a garbling scheme, which ensures the correctness of the garbling process by the verification algorithm. Jumping ahead, this guarantees the effectiveness of witness extraction given two accepting transcripts.

Definition 5 (Verifiability). *Let $\mathcal{G}$ be a garbling scheme described as above. We say that $\mathcal{G}$ enjoys verifiability if for all $C : \{0,1\}^n \to \{0,1\}$, for all PPT $\mathcal{A}$, the following probability is negligible in parameter κ.*

$$\Pr\left[\begin{matrix}(GC, e, d, W) \leftarrow \mathcal{A}(1^\kappa, C) \\ \mathsf{Ve}(C, GC, e) = 1 \wedge \mathsf{De}(d, \mathsf{Ev}(GC, W)) = 1\end{matrix} : \mathsf{ev}(C, \mathsf{Gb.Ext}(e, W)) \neq 1\right]$$

We note that the above definition differs from the verifiability definition in [20]. Nevertheless, we show that under the assumption that the encoding function is injective and projective (i.e., efficient extraction is possible), Definition 5 is implied by the original one, which we recall below.

```
Gb.Ext$(e, W)$

for $i := 1$ to $n$ do
  if $W_i = \mathsf{En}_i(e, 0)$ then
    $w_i := 0$
  elseif $W_i = \mathsf{En}_i(e, 1)$ then
    $w_i := 1$
  else
    return $\perp$
  endif
endfor
return $w_1, ..., w_n$
```

Fig. 2. The input extraction procedure of an injective and projective garbling scheme $\mathcal{G}$. n is the input length and En_i is the i^{th} output block of En.

Definition 6 (Verifiability from [20]). *A garbling scheme $\mathcal{G}$ enjoys verifiability if for all $C : \{0,1\}^n \to \{0,1\}$ and $x, y \in \{0,1\}^n$, for all PPT $\mathcal{A}$, the following probability is negligible in parameter κ.*

$$\Pr\left[\begin{matrix}(GC, e) \leftarrow \mathcal{A}(1^\kappa, C) \\ X = \mathit{En}(e, x), Y = \mathit{En}(e, y)\end{matrix} : \begin{matrix}\mathit{Ve}(C, GC, e) = 1 \quad \wedge \\ \mathit{Ev}(GC, X) \neq \mathit{Ev}(GC, Y)\end{matrix}\right]$$

Lemma 1. *Let $\mathcal{G}$ be a garbling scheme and the encoding function En is injective and projective, then Definition 6 implies Definition 5.*

Proof. Suppose a garbling scheme $\mathcal{G}$ satisfies Definition 6 but does not satisfy Definition 5. Consider the adversary $\mathcal{A}$ that returns (GC, e, d, W) on input C. The encoded input W satisfies that $\mathsf{De}(d, \mathsf{Ev}(GC, W)) = 1$ while the value returned from the extraction procedure $\mathsf{Gb.Ext}$ — denoted as w — evalautes to 0 on C.[1]

Let $W' = \mathsf{En}(e, w)$. From the correctness and the deterministic property of the decoding procedure we conclude that $\mathsf{Ev}(GC, W) \neq \mathsf{Ev}(GC, W')$. Since all other requirements in Definition 6 are met, this forms a contradiction.

3 Construction

In this section, we present our construction of a public-coin zero-knowledge proof system based on garbled circuits.

[1] We ignore the case of $\mathsf{Gb.Ext}$ returning "$\perp$" since the *authenticity* property of the garbling scheme (which is standard and not presented in this paper) guarantees that an adversary cannot generate such malformed encoded input that evaluates to well-formed encoded output.

3.1 ZK in the Standard Model

In contrast to the well-known JKO protocol [20], we let the prover perform garbling in order to acquire a public-coin protocol. The protocol, which is shown in Figure 3, is a cut-and-choose style sigma protocol where the prover first sends two garblings of a circuit C_x using independent random coins. When considering an $\mathcal{NP}$ relation $\mathcal{R}$ such that $(x, w) \in \mathcal{R}$ iff. $C(x, w) = 1$, we let the circuit C_x "hard-wire" the public information x, i.e., $C_x(w) = C(x, w)$.

Then the verifier samples a challenge $b \leftarrow \{0, 1\}$ and sends it to the prover. The prover reveals the coins specified by this index and sends input encodings W corresponding to the unopened circuit. Finally, the verifier accepts the proof if the random coin r_b successfully generates GC_b and the outputs induced by GC_{1-b} and X is 1.

Prover		**Verifier**
$(GC_0, e_0, d_0) \leftarrow \mathsf{Gb}(1^\kappa, C_x)$		
$(GC_1, e_1, d_1) \leftarrow \mathsf{Gb}(1^\kappa, C_x)$		
	$\xrightarrow{GC_0, d_0, GC_1, d_1}$	
		$b \leftarrow \{0, 1\}$
	$\xleftarrow{b}$	
$W \leftarrow \mathsf{En}(e_{1-b}, w)$		
	$\xrightarrow{e_b, W}$	
		The verifier accepts if
		$\mathsf{Ve}(GC_b, e_b, C_x) = 1 \ \wedge$
		$\mathsf{De}(d_{1-b}, \mathsf{Ev}(GC_{1-b}, W)) = 1$

Fig. 3. A public coin zero-knowledge in the standard model.

We prove the special soundness and honest-verifier zero-knowledge properties of this scheme in the following theorem.

Theorem 1. *Let $\mathcal{R}$ be a NP relation defined by circuit family $\{C'\}$ s.t. for every instance $(a, b) \in \mathcal{R}, C'(a, b) = 1$. For any instance $a \in L(\mathcal{R})$, define circuit family $C(w) = C'(a, w)$. Let $\mathcal{G} = (\mathsf{Gb}, \mathsf{En}, \mathsf{De}, \mathsf{Ev}, \mathsf{ev}, \mathsf{Ve})$ be a garbling scheme. The sigma protocol in Figure 3 is a computational honest-verifier zero-knowledge protocol with special soundness for the relation $\mathcal{R}$.*

Proof. The completeness of the protocol follows from the correctness of the garbling scheme. We then prove the honest-verifier zero-knowledge property by explaining the procedure for generating an accepting verifier's transcript.

The procedure for generating an accepting transcript from input (x, b) is as follows:

- First generate $(GC_b, e_b, d_b) \leftarrow \mathsf{Gb}(1^\kappa, C_x)$ as an honest prover.
- Then generate the other garbled circuit and input by invoking the simulation algorithm for the garbling scheme: $(GC_{1-b}, W, d_{1-b}) \leftarrow \mathsf{Gb.Sim}(C_x, 1)$.
- Outputs the transcript $(GC_0, d_0, GC_1, d_1, b, e_b, W)$.

The effectiveness of the simulator $\mathsf{Gb.Sim}$ for the garbling scheme, which is implied by the privacy of the garbling scheme (Definition 4), guarantees the computational indistinguishability between a real accepting transcript and one generated from the above procedure (without the knowledge of input w), which implies computational zero-knowledge.

Next we argue the special soundness of the above scheme. Recall that this property requires that a valid input can be extracted from two transcripts (a, e, z) and (a, e', z') such that $e \neq e'$. Consider the two transcripts

- $(GC_0, d_0, GC_1, d_1, b, e_b, W)$
- $(GC_0, d_0, GC_1, d_1, b', e_{b'}, W')$

where without loss of generality we many assume $b = 0$ and $b' = 1$. Notice that for both indices, the condition $\mathsf{Ve}(GC, e, d, C_x) = 1 \wedge \mathsf{De}(d, \mathsf{Ev}(GC, W))$ holds. From the verifiability property (Definition 5) of the garbling scheme, we conclude that with non-negligible probability we can extract input w such that $\mathsf{ev}(C_x, w) = 1$ for either transcript.

We note that the above proof does not require rewinding and thus in the quantum random oracle model (QROM) the security properties hold against quantum adversaries [10,23]. □

Next, we apply the standard techniques, namely parallel repetition and Fiat-Shamir transform, to the basic sigma protocol in Figure 3, in order to acquire a non-interactive zero-knowledge proof with negligible soundness error in the random oracle model.

Corollary 1. *Let H be a random oracle and $\lambda \in \mathbb{N}$ be an integer. Then by running the protocol in Figure 3 for λ times in parallel and generating each challenge by hashing all κ first messages using the random oracle H, one can acquire a non-interactive zero-knowledge proof with honest-verifier zero-knowledge and soundness error $2^{-\lambda}$ against PPT adversary.*

Optimizations. Notice that in our protocol a large overhead originates from the application of the cut-and-choose technique. Indeed, to achieve soundness error $2^{-\kappa}$, the prover needs to send 2κ garbled circuits where only one is actually evaluated. And therefore it is natural to apply the optimizations targeted at the cut-and-choose technique, commonly found in the malicious two-party computation setting.

– Instead of sending the garbled circuit, the prover can only send *commitments* in the first round. Then in the third round, the prover sends the coins for garbling and commitment for the selected index and decommitment information for the other index. This can reduce the communication complexity roughly by half.

Recall that a large overhead of the protocol in this subsection is caused by the need to verify that the prover faithfully garbles the correct circuit. Notwithstanding, a trusted party (e.g., a judiciary department) may exist in some scenarios, and can distribute some input-independent "raw-material" to both parties before the actual proof process begins. This is captured by the "Common Reference String" model in the next subsection.

3.2 ZK in the CRS Model

In this setting, a third party garbles the verification circuit and distributes the input encoding information to the prover and garbled circuit to the verifier faithfully. Notice that since the garbled circuit is guaranteed to be correct by the model, we can remove the expensive cut-and-choose step, and the proof message consists of only the garbled input (which is trivially non-interactive). This is captured in the algorithms in Figure 4.

```
CRS.Gen($1^n$)
    $(GC, e, d) \leftarrow \mathsf{Gb}(C, 1^n)$
    return $((GC, d), e)$
```

```
Prove($e, x$)
    $X \leftarrow \mathsf{En}(e, x)$
    return $X$
```

```
Verify($GC, d, X$)
    $y \leftarrow \mathsf{De}(d, \mathsf{En}(GC, X))$
    return $y$
```

Fig. 4. Zero-knowledge protocol in the common reference string model.

3.3 Discussion

Recall that in the construction we proposed, the prover's computation is essentially garbling and encoding. From a theoretical perspective, this paradigm can be viewed as an application of the randomized encoding technique [2], where the prover essentially performs the encoder's job. When instantiated with the garbled circuit, the prover's computation is in $\mathcal{NC}1$ — the class of functions that can be completed by a $O(\log n)$-depth circuit family. This characterization could inspire further applications, such as utilizing parallelism or delegation of computation.

4 Implementation and Experiments

We implement the protocol in Section 3 for the `SHA256` relation defined as follows:

$$\mathcal{R}^{\text{hash}}(y;x) : y = \texttt{SHA256}(x), \text{ where } x \in \{0,1\}^{512}, \text{ and } y \in \{0,1\}^{256}.$$

Throughout the experiment, we use the specific `SHA256` circuit in the works of Campanelli et al. [8] which has optimized the `AND` gate count. The total number of gates is 117,016 and the number of `AND` gates is 22,272.

To counter the Grover quantum attack, we extend the size of a garbled label to 256bit. To achieve 2^{-128} soundness error, the repetition count is 128.

The proving time is 3.0s. The verification time is 2.2s. The proof is 379MB.

We run the experiments on a Ubuntu 20.04 LTS machine with AMD Ryzen 5 3600 CPU and 16GB of RAM. Our implementation is only 728 lines in C++ with dependency on OpenSSL.

5 Conclusion

We admit the proof size of our construction is large. However, in comparison with other post-quantum NIZK, our construction requires minimum knowledge. This scheme can be taught to undergraduate students right after they understand garbled circuits. Unlike zk-STARK, which requires plenty of efforts on complexity theory, or MPC-in-the-head paradigm, which requires secure multiparty computation in advance. Our implementation is only 728 lines, which is suitable as a course work for beginners.

Second, our construction is highly parallel. Not only cut-and-choose can be parallel, but also garbling itself. That means the execution time can be reduced by multiprocessor significantly.

On the other hand, our construction benefits from the improvements on garbled circuits. For example, Heath et al. recently purpose stacked garbling [18], indicating that our approach works potentially better than others on some specific tasks like evaluating a decision tree.

References

1. Ames, S., Hazay, C., Ishai, Y., Venkitasubramaniam, M.: Ligero: Lightweight sublinear arguments without a trusted setup. In: Thuraisingham et al. [26], pp. 2087–2104. https://doi.org/10.1145/3133956.3134104
2. Applebaum, B.: Garbled circuits as randomized encodings of functions: a primer. Cryptology ePrint Archive, Report 2017/385 (2017), http://eprint.iacr.org/2017/385
3. Beaver, D., Micali, S., Rogaway, P.: The round complexity of secure protocols. In: Proceedings of the twenty-second annual ACM symposium on Theory of computing. pp. 503–513 (1990)
4. Bellare, M., Hoang, V.T., Rogaway, P.: Foundations of garbled circuits. In: Yu, T., Danezis, G., Gligor, V.D. (eds.) ACM CCS 2012. pp. 784–796. ACM Press (Oct 2012). https://doi.org/10.1145/2382196.2382279
5. Ben-Sasson, E., Bentov, I., Horesh, Y., Riabzev, M.: Scalable, transparent, and post-quantum secure computational integrity. Cryptology ePrint Archive, Report 2018/046 (2018), https://eprint.iacr.org/2018/046

6. Boldyreva, A., Micciancio, D. (eds.): CRYPTO 2019, Part II, LNCS, vol. 11693. Springer, Heidelberg (Aug 2019)
7. Bootle, J., Cerulli, A., Chaidos, P., Groth, J., Petit, C.: Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. In: Fischlin, M., Coron, J.S. (eds.) EUROCRYPT 2016, Part II. LNCS, vol. 9666, pp. 327–357. Springer, Heidelberg (May 2016). https://doi.org/10.1007/978-3-662-49896-5_12
8. Campanelli, M., Gennaro, R., Goldfeder, S., Nizzardo, L.: Zero-knowledge contingent payments revisited: Attacks and payments for services. In: Thuraisingham et al. [26], pp. 229–243. https://doi.org/10.1145/3133956.3134060
9. Chase, M., Derler, D., Goldfeder, S., Orlandi, C., Ramacher, S., Rechberger, C., Slamanig, D., Zaverucha, G.: Post-quantum zero-knowledge and signatures from symmetric-key primitives. In: Thuraisingham et al. [26], pp. 1825–1842. https://doi.org/10.1145/3133956.3133997
10. Don, J., Fehr, S., Majenz, C., Schaffner, C.: Security of the Fiat-Shamir transformation in the quantum random-oracle model. In: Boldyreva and Micciancio [6], pp. 356–383. https://doi.org/10.1007/978-3-030-26951-7_13
11. Gennaro, R., Gentry, C., Parno, B., Raykova, M.: Quadratic span programs and succinct NIZKs without PCPs. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 626–645. Springer, Heidelberg (May 2013). https://doi.org/10.1007/978-3-642-38348-9_37
12. Giacomelli, I., Madsen, J., Orlandi, C.: ZKBoo: Faster zero-knowledge for Boolean circuits. In: Holz, T., Savage, S. (eds.) USENIX Security 2016. pp. 1069–1083. USENIX Association (Aug 2016)
13. Glaser, A., Barak, B., Goldston, R.J.: A zero-knowledge protocol for nuclear warhead verification. Nature **510**(7506), 497–502 (2014)
14. Goldreich, O.: Foundations of Cryptography: Basic Applications, vol. 2. Cambridge University Press, Cambridge, UK (2004)
15. Goldwasser, S., Kalai, Y.T., Rothblum, G.N.: Delegating computation: interactive proofs for muggles. In: Ladner, R.E., Dwork, C. (eds.) 40th ACM STOC. pp. 113–122. ACM Press (May 2008). https://doi.org/10.1145/1374376.1374396
16. Groth, J.: Short pairing-based non-interactive zero-knowledge arguments. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 321–340. Springer, Heidelberg (Dec 2010). https://doi.org/10.1007/978-3-642-17373-8_19
17. Hazay, C., Lindell, Y.: Efficient Secure Two-Party Protocols - Techniques and Constructions. ISC, Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14303-8
18. Heath, D., Kolesnikov, V.: Stacked garbling - garbled circuit proportional to longest execution path. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020, Part II. LNCS, vol. 12171, pp. 763–792. Springer, Heidelberg (Aug 2020). https://doi.org/10.1007/978-3-030-56880-1_27
19. Ishai, Y., Kushilevitz, E., Ostrovsky, R., Sahai, A.: Zero-knowledge from secure multiparty computation. In: Johnson, D.S., Feige, U. (eds.) 39th ACM STOC. pp. 21–30. ACM Press (Jun 2007). https://doi.org/10.1145/1250790.1250794
20. Jawurek, M., Kerschbaum, F., Orlandi, C.: Zero-knowledge using garbled circuits: how to prove non-algebraic statements efficiently. In: Sadeghi, A.R., Gligor, V.D., Yung, M. (eds.) ACM CCS 2013. pp. 955–966. ACM Press (Nov 2013). https://doi.org/10.1145/2508859.2516662
21. Kilian, J.: A note on efficient zero-knowledge proofs and arguments (extended abstract). In: 24th ACM STOC. pp. 723–732. ACM Press (May 1992). https://doi.org/10.1145/129712.129782

22. Kolesnikov, V., Schneider, T.: Improved garbled circuit: Free XOR gates and applications. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part II. LNCS, vol. 5126, pp. 486–498. Springer, Heidelberg (Jul 2008). https://doi.org/10.1007/978-3-540-70583-3_40
23. Liu, Q., Zhandry, M.: Revisiting post-quantum Fiat-Shamir. In: Boldyreva and Micciancio [6], pp. 326–355. https://doi.org/10.1007/978-3-030-26951-7_12
24. Malkhi, D., Nisan, N., Pinkas, B., Sella, Y.: Fairplay - secure two-party computation system. In: Blaze, M. (ed.) USENIX Security 2004. pp. 287–302. USENIX Association (Aug 2004)
25. Micali, S.: CS proofs (extended abstracts). In: 35th FOCS. pp. 436–453. IEEE Computer Society Press (Nov 1994). https://doi.org/10.1109/SFCS.1994.365746
26. Thuraisingham, B.M., Evans, D., Malkin, T., Xu, D. (eds.): ACM CCS 2017. ACM Press (Oct / Nov 2017)
27. Xie, T., Zhang, J., Zhang, Y., Papamanthou, C., Song, D.: Libra: Succinct zero-knowledge proofs with optimal prover computation. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019, Part III. LNCS, vol. 11694, pp. 733–764. Springer, Heidelberg (Aug 2019). https://doi.org/10.1007/978-3-030-26954-8_24
28. Yao, A.C.C.: How to generate and exchange secrets (extended abstract). In: 27th FOCS. pp. 162–167. IEEE Computer Society Press (Oct 1986). https://doi.org/10.1109/SFCS.1986.25
29. Zahur, S., Rosulek, M., Evans, D.: Two halves make a whole - reducing data transfer in garbled circuits using half gates. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015, Part II. LNCS, vol. 9057, pp. 220–250. Springer, Heidelberg (Apr 2015). https://doi.org/10.1007/978-3-662-46803-6_8
30. Zhang, J., Xie, T., Zhang, Y., Song, D.: Transparent polynomial delegation and its applications to zero knowledge proof. In: 2020 IEEE Symposium on Security and Privacy. pp. 859–876. IEEE Computer Society Press (May 2020). https://doi.org/10.1109/SP40000.2020.00052